NASA Technical Memorandum 86711

The NAS Kernel Benchmark Program

David H. Bailey and John T. Barton

August 1985

NASA
National Aeronautics and
Space Administration

NASA Technical Memorandum 86711

The NAS Kernel Benchmark Program

David H. Bailey, Informatics General Corporation, Palo Alto, California
John T. Barton, Ames Research Center, Moffett Field, California

August 1985

NASA
National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

THE NAS KERNEL BENCHMARK PROGRAM

David H. Bailey[1] and John T. Barton[2]

Ames Research Center

SUMMARY

A benchmark test program that measures supercomputer performance has been developed for the use of the NAS (Numerical Aerodynamic Simulation) Projects Office at NASA Ames Research Center. This benchmark program is described in detail and the specific ground rules for running the program as a performance test are discussed.

[1]Employee of Informatics General Corp., Palo Alto, California, under contract NAS2-11555.

[2]Employee of NASA Ames Research Center in the NAS Projects Office.

INTRODUCTION

A benchmark test program has been developed for use by the NAS program at NASA Ames Research Center to aid in the evaluation of supercomputer performance. This program consists of seven Fortran test kernels that perform calculations that are typical of Ames supercomputing. It is expected that the performance of a supercomputer system on this program will provide an accurate projection of the performance of the system on actual NAS program computer codes. This paper describes the test program in detail and lists the specific ground rules that have been established for running the program as a performance test.

PROGRAM DESCRIPTION

The NAS Kernel Benchmark Program consists of approximately 1000 lines of Fortran code, organized into seven separate tests. Each individual test consists of a loop that iteratively calls a certain subroutine. These subroutines were chosen after review of many of the calculations currently being performed on Ames supercomputers and by recommendations from a number of Ames scientists and programmers, particularly those working on computational fluid dynamics problems. In most cases, these subroutines have been extracted from actual programs currently in use, and they have been incorporated into the NAS Kernel Benchmark Program with only minor changes. Thus it is felt that these test kernels are a representative cross section of expected NAS program supercomputing, and the performance of a computer system (both its hardware and its Fortran compiler) on these tests should be a reliable predictor of the actual system performance on NAS user programs.

The seven selected programs all emphasize the vector performance of a computer system. Almost all of the floating-point operations indicated in these Fortran subroutines are contained in loops that are computable by vector operations, provided that the Fortran compiler of the computer system being tested is sufficiently powerful in its vectorization analysis, and provided that the hardware design of the computer includes the necessary vector instructions. Most serious supercomputer programs currently in use at Ames are fairly highly vectorized, and it is expected that programs to be developed in the future will virtually all be designed to effectively use the vector processing capabilities of supercomputers. Some programs that have substantial scalar processing will continue to be used, but it is expected that their numbers will decline as algorithms and codes that are more suitable for vector processing are developed. Another reason for emphasizing vector performance in these benchmark kernels is that it is not very meaningful to average, even in a harmonic average sense, the performance of a supercomputer on a scalar code with its performance on a vector code.

This program not only tests the hardware execution speed of a computer, but it also tests the effectiveness of the Fortran compiler. It is clear that a phenomenally fast hardware design is worthless unless it is coupled with a Fortran compiler that can fully

utilize the advanced hardware design. Furthermore, it is becoming increasingly clear that vectorization and other optimizations must either be completely automatic or be very easy to direct. If effective utilization of a computer requires massive redesign of otherwise well-written, standard Fortran-77 code, or if a high level of performance is possible only by considerable human intervention, then the actual usable power of the computer is severely reduced.

The seven test kernels of the NAS Kernel Benchmark Program have, for the most part, been developed quite recently. As a result, they represent Fortran programs that have been designed and written for modern vector computation, as opposed to the somewhat dated code that is used for other popular benchmark programs. It might be argued that there is some inherent bias in the test towards the Cray computers, since most of these kernels were written on a Cray X-MP. However, substantial care was exercised in the selection of these kernels to insure that none of them had any constructs that would unduly favor the Cray line. As much as possible, subroutines were selected that were merely straightforward Fortran code, intelligently coded with loops that are capable of being executed with vector operations, but otherwise neutral towards any particular machine. In fact, in the process of selecting these kernels for testing, it was discovered that some of them actually caused unforeseen difficulties for the Cray compiler. Nevertheless, they were left in the test suite to maintain objectivity.

Performance is measured by the NAS Kernel Benchmark Program in MFLOPS (millions of floating-point operations per second). The precise number of floating-point operations for the various functions used in the test kernels is shown in Table 1. These numbers are based on actual counts of 64-bit floating-point operations in published algorithms.

It should be noted that this program only measures MFLOPS rates. Disk I/O, operating system efficiency, and other important factors of overall performance are not measured by this benchmark program. Also, several of the test subroutines perform a significant amount of memory move, integer, and logical operations, none of which is included in the floating-point operation count.

The following is a description of the seven proposed Fortran test kernels. Other features are summarized in Table 2.

1. MXM – This subroutine performs the usual matrix product on two input matrices. The subroutine employs a four-way unrolled, outer product matrix multiply algorithm that is especially effective for most vector computers. See [1] for a discussion of this algorithm.
2. CFFT2D – This test performs a complex radix 2 FFT on a two dimensional input array, returning the result in place. The test kernel actually consists of two subroutines that perform FFTs along the first and second dimension of the array, respectively, taking advantage of the parallel structure of the array. See [2] for a discussion of the FFT algorithm used.
3. CHOLSKY – This subroutine performs a Cholesky decomposition in parallel on

Table 1: Floating-point Operation Counts

FIRST ARGUMENT	FUNCTION	SECOND ARGUMENT	FLOATING PT. OPS.
Real	+	Real	1
Real	-	Real	1
Real	*	Real	1
1	/	Real	2
Real	/	Real	3
Real	**	2	1
Real	**	Real	45
Complex	*	Real	2
Complex	/	Real	4
1	/	Complex	7
Real	/	Complex	9
Complex	+	Complex	2
Complex	-	Complex	2
Complex	*	Complex	6
Complex	/	Complex	13
Real	SQRT		12
Real	EXP		18
Real	LOG		25
Real	SIN		25
Real	ATAN		25
Complex	ABS		15
Complex	EXP		70
Complex	LOG		65

Table 2: Kernel Features

FEATURE	KERNEL 1	2	3	4	5	6	7
Two dimensional arrays	X	X			X	X	X
Multidimensional arrays			X	X			X
Dimensions with colons			X				
Integer arrays		X			X	X	
Integer functions in indices					X	X	
IF statements in inner loops						X	
Scientific function calls		X	X		X	X	
Complex arithmetic		X			X	X	
Complex function calls					X	X	
Inner loop memory strides	1	1	1	1	1	1	128
		2	4	2	2		
		256		750	500		
				900			
Inner loop vector lengths	256	128	250	28	5	100	128
		256			100	500	
					500	1000	

a set of input matrices, which are actually input to the subroutine as a single three-dimensional array.

4. BTRIX – This kernel performs a block tridiagonal matrix solution along one dimension of a four dimensional array.
5. GMTRY – This subroutine sets up arrays for a vortex method solution and performs Gaussian elimination on the resulting array. This kernel is noted for a number of loops that are challenging to vectorize.
6. EMIT – Also extracted from a vortex code, this subroutine creates new vortices according to certain boundary conditions.
7. VPENTA – This subroutine simultaneously inverts three matrix pentadiagonals in a highly parallel fashion.

In each of the above test subroutines, the input data arrays are filled by a portable pseudorandom number generator in the calling program. This feature insures that all computers running the NAS Kernel Benchmark Program will perform the required calculations on the same numbers. It also permits the output results to be checked for accuracy. Each of the seven tests is independent from the others – none depends on results calculated in a previous test program. Thus program alterations to improve the execution speed of one of the test kernels may be made without fear of affecting the other kernels.

GROUND RULES FOR PERFORMANCE TESTING

Worlton's recent article [3] pointed out some of the difficulties that are involved in supercomputer performance testing. Most of these problems are a result of the lack of well-defined controls on these tests. For instance, in some recent test results, one vendor was apparently allowed to perform some minor tuning and insertion of compiler directives, whereas the other was not. In other cases confusion has resulted from researchers not carefully noting exactly which version of a vendor's compiler was being used in their tests. Some vendors have claimed amazingly high performance rates for their computers, which, upon closer analysis, have been achieved only by massive recoding of the test kernels and by the usage of assembly code. As a result of these difficulties, many of the recent comparisons of supercomputer performance have degenerated into shouting matches that have generated more heat than light.

In consideration of such problems, some strict ground rules have been established for using the NAS Kernel Benchmark Program to evaluate supercomputer performance. Also, four levels of tests have been defined, so that the effects of varying amounts of tuning may be assessed. These different levels will also enable the NAS program to differentiate the performance of the hardware from that of the compiler. If the compiler is truly effective, then a relatively small amount of tuning should be sufficient to achieve close to the full potential of the hardware. The four test levels are defined as follows:

1. Level 0 ("dusty deck"): For this test, the NAS Kernel Benchmark Program must be run without any changes to improve performance. If any alterations are required for compatibility purposes (for example, to define the timing function), they must be made by NAS program personnel.
2. Level 20 ("minor tuning"): For this test, a few minor alterations may be made to the code to enhance performance. These changes may include, for example, compiler directives to assist the compiler's vectorization analysis or changes to array dimensions to avoid disadvantageous memory strides. No more than 20 lines of code in the entire program file may be inserted or modified.
3. Level 50 ("major tuning"): For this test, more extensive modifications may be made to the code to enhance performance. For example, some loops may be rewritten to avoid constructs that cause difficulties for the compiler or the hardware. A total of up to 50 lines of the program file may be inserted or modified for this test.
4. Level 1000 ("customized code"): For this test, large scale coding changes are allowed to improve performance. Entire subroutines may be rewritten to avoid difficult constructs. There is no limit to the number of lines of code that may be inserted or modified.

For all four levels of tests, any modifications made to the program code must conform

to the ANSI Fortran-77 standard [4]. In particular, absolutely no assembly code will be allowed within the program file, and no external programs may be referenced other than the standard Fortran functions. Fortran subprograms may be referenced only if the Fortran code for the subprograms is included in the program file and conforms to the other requirements mentioned in this paper. Finally, no modification to the algorithms in the code may change the number of floating-point operations performed.

The precision level of all floating-point data and operations in the program must be 64 bits, with at least 47 mantissa bits. As a test of the hardware precision, and to ensure that any modifications made to the program file have not fundamentally changed the calculations being performed, an accuracy check is included with each of the seven tests. These checks are performed by comparing a selected result from each of the programs with a reference value stored in the program code and then computing the fractional error. The total of the fractional errors from the seven programs must be less than 5×10^{-10}.

The NAS Kernel Benchmark Program automatically calculates performance statistics and outputs this report on Fortran unit 6. This report includes the results of the accuracy checks, the number of floating-point operations performed, the CPU run times, and the resulting MFLOPS rates. The total error, total floating-point operation count, total CPU time, and the overall MFLOPS rate are also included.

Normally only uniprocessor results are tabulated. If desired, multiprocessor performance may be estimated by simultaneously running the benchmark program on each of the individual processors. A multiprocessing performance figure may then computed by averaging the timings from the runs on the individual processors. Although no explicit multiprocessing is performed in this manner, such an exercise measures the amount of interprocessor resource contention, which is a significant factor in multiprocessing. In this way the performance increase that can be expected from multiple processor computation can be estimated without making the laborious modifications that are usually required to invoke true multiprocessing.

REFERENCES

1. Hockney, R. W., and Jesshope, C. R., *Parallel Computers*, Adam Hilger, Bristol, England, 1981.
2. Brigham, E. Oran, *The Fast Fourier Transform*, Prentice-Hall, Englewood Cliffs, N.J., 1974.
3. Worlton, Jack, "Understanding Supercomputing Benchmarks", Datamation, September 1, 1984, p. 121.
4. American National Standards Institute, *ANSI Fortran X3.9-1978*, ANSI, New York, 1978.

APPENDIX:

PROGRAM LISTING

```
      PROGRAM NASKER
C
C   NAS KERNEL BENCHMARK PROGRAM
C   12/17/84      DAVID H BAILEY
C
      CHARACTER*8 PN(8)
      REAL ER(8), FP(8), TM(8), RT(8)
      COMMON /ARRAYS/ DATA(360000)
      DATA PN/'MXM', 'CFFT2D', 'CHOLSKY', 'BTRIX', 'GMTRY', 'EMIT',
     $ 'VPENTA', 'TOTAL'/
C
      WRITE (6, 1)
1     FORMAT (/16X, 'THE NAS KERNEL BENCHMARK PROGRAM'//)
C
      CALL MXMTST (ER(1), FP(1), TM(1))
      CALL FFTTST (ER(2), FP(2), TM(2))
      CALL CHOTST (ER(3), FP(3), TM(3))
      CALL BTRTST (ER(4), FP(4), TM(4))
      CALL GMTTST (ER(5), FP(5), TM(5))
      CALL EMITST (ER(6), FP(6), TM(6))
      CALL VPETST (ER(7), FP(7), TM(7))
C
      TE = 0.
      TF = 0.
      TT = 0.
      DO 100 I = 1, 7
        TE = TE + ER(I)
        TF = TF + FP(I)
        TT = TT + TM(I)
        RT(I) = 1E-6 * FP(I) / TM(I)
100   CONTINUE
      ER(8) = TE
      FP(8) = TF
      TM(8) = TT
      RT(8) = 1E-6 * TF / TT
C
      WRITE (6, 2)  (PN(I), ER(I), FP(I), TM(I), RT(I), I = 1, 8)
2     FORMAT (' PROGRAM', 8X, 'ERROR', 10X, 'FP OPS', 7X, 'SECONDS',
     $  6X, 'MFLOPS'// 7(1X, A8, 1P2E15.4, 0PF12.4, F12.2)//
     $  1X, A8, 1P2E15.4, 0PF12.4, F12.2//)
C
      STOP
      END
C
      FUNCTION CPTIME ()
C
C   RETURNS THE CPU TIME SINCE THE LAST CALL TO CPTIME.
C   THIS SUBPROGRAM MAY BE CHANGED AS NEEDED FOR A PARTICULAR COMPUTER
C   SYSTEM WITHOUT PENALTY, PROVIDED IT PERFORMS THIS FUNCTION.
C
      DATA TX/0./
      T = SECOND ()
      CPTIME = T - TX
      TX = T
      RETURN
      END
C
      SUBROUTINE COPY (N, A, B)
C
C   ARRAY COPY ROUTINE
C
      REAL A(N), B(N)
      DO 100 I = 1, N
        B(I) = A(I)
100   CONTINUE
      RETURN
      END
C
      SUBROUTINE MXMTST (ER, FP, TM)
C
C   FLOATING-POINT MATRIX MULTIPLY TEST
C
      PARAMETER (L=256, M=128, N=64, F7=78125., T30=1073741824.)
      COMMON /ARRAYS/ A(L,M), S1, B(M,N), S2, C(L,N)
      DATA IT/100/, ANS/35.2026179738722/
C
C   INITIALIZATION
C
C   THE ARRAYS A AND B ARE FILLED WITH PSEUDO-RANDOM  (0., 1.)  DATA
C   USING A RANDOM NUMBER GENERATOR BASED ON THE RECURSION
C         X(N+1) = 5**7 * X(N)   (MOD 2**30)
C   THIS RECURSION WILL GENERATE 2**28 (APPROX. 268 MILLION) NUMBERS
C   BEFORE REPEATING.  FOR THIS SCHEME TO WORK PROPERLY, THE HARDWARE
C   MULTIPLY OPERATION MUST BE CORRECT TO 47 BITS OF PRECISION.
C   THIS SAME SCHEME IS USED TO INITIALIZE DATA ARRAYS FOR ALL TESTS.
C
      T = F7 / T30
      DO 100 J = 1, M
        DO 100 I = 1, L
          T = MOD (F7 * T, 1.)
          A(I,J) = T
100   CONTINUE
      DO 110 J = 1, N
        DO 110 I = 1, M
          T = MOD (F7 * T, 1.)
          B(I,J) = T
110   CONTINUE
      TM = CPTIME ()
C
C   TIMING TEST
C
      DO 120 II = 1, IT
        CALL MXM (A, B, C, L, M, N)
120   CONTINUE
C
      TM = CPTIME ()
      ER = ABS ((C(19,19) - ANS) / ANS)
      FP = 2. * IT * L * M * N
C
      RETURN
      END
C
      SUBROUTINE MXM (A, B, C, L, M, N)
      DIMENSION A(L,M), B(M,N), C(L,N)
C
C   4-WAY UNROLLED MATRIX MULTIPLY ROUTINE FOR VECTOR COMPUTERS.
C   M MUST BE A MULTIPLE OF 4.  CONTIGUOUS DATA ASSUMED.
C   D H BAILEY  11/15/84
C
      DO 100 K = 1, N
        DO 100 I = 1, L
          C(I,K) = 0.
100     CONTINUE
      DO 110 J = 1, M, 4
        DO 110 K = 1, N
          DO 110 I = 1, L
            C(I,K) = C(I,K) + A(I,J) * B(J,K)
     $             + A(I,J+1) * B(J+1,K) + A(I,J+2) * B(J+2,K)
     $             + A(I,J+3) * B(J+3,K)
110   CONTINUE
C
      RETURN
      END
C
      SUBROUTINE FFTTST (ER, FP, TM)
C
C   2-D FFT TEST PROGRAM
C
      PARAMETER (M=128, N=256, M1=128, F7=78125., T30=1073741824.)
      COMPLEX X, Y, CT
      COMMON /ARRAYS/ X(M1,N), W1(M), W2(N), IP(2*N)
      DATA IT/100/, ANS/0.894799941219277/
C
C   INITIALIZE
C
      AMN = M * N
```

```
      RMN = 1. / AMN
      T2 = F7 / T30
      DO 100 J = 1, N
        DO 100 I = 1, M
          T1 = MOD (F7 * T2, 1.)
          T2 = MOD (F7 * T1, 1.)
          X(I,J) = CMPLX (T1, T2)
100   CONTINUE
      CALL CFFT2D1 (0, M, M1, N, X, W1, IP)
      CALL CFFT2D2 (0, M, M1, N, X, W2, IP)
      TM = CPTIME ()
C
C     TEST ITERATIONS
C
      DO 120 K = 1, IT
        DO 110 J = 1, N
          DO 110 I = 1, M
            X(I,J) = RMN * X(I,J)
110     CONTINUE
C
        CALL CFFT2D1 (1, M, M1, N, X, W1, IP)
        CALL CFFT2D2 (1, M, M1, N, X, W2, IP)
        CALL CFFT2D2 (-1, M, M1, N, X, W2, IP)
        CALL CFFT2D1 (-1, M, M1, N, X, W1, IP)
120   CONTINUE
C
      TM = CPTIME () 
      ER = ABS ((REAL(X(19,19)) - ANS) / ANS)
      FP = IT * AMN * (2.  +  10. * LOG (AMN)/LOG (2.))
C
      RETURN
      END
C
      SUBROUTINE CFFT2D1 (IS, M, M1, N, X, W, IP)
C
C     PERFORMS COMPLEX RADIX 2 FFTS ON THE FIRST DIMENSION OF THE 2-D ARRAY X
C     D H BAILEY  11/15/84
C
      COMPLEX X(M1,N), W(M), CT, CX
      INTEGER IP(2,M)
      DATA PI/3.14159265358979 3/
C
C     IF  IS = 0  THEN INITIALIZE ONLY
C
      M2 = M / 2
      IF (IS .EQ. 0)  THEN
        DO 100 I = 1, M2
          T = 2. * PI * (I-1) / M
          W(I) = CMPLX (COS (T), SIN (T))
100     CONTINUE
        RETURN
      ENDIF
C
C     PERFORM FORWARD OR BACKWARD FFTS ACCORDING TO  IS = 1 OR -1
C
      DO 110 I = 1, M
        IP(1,I) = I
110   CONTINUE
      L = 1
      I1 = 1
C
120   I2 = 3 - I1
      DO 130 J = L, M2, L
        CX = W(J-L+1)
        IF (IS .LT. 0)  CX = CONJG (CX)
        DO 130 I = J-L+1, J
          II = IP(I1,I)
          IP(I2,I+J-L) = II
          IM = IP(I1,I+M2)
          IP(I2,I+J) = IM
          DO 130 K = 1, N
            CT = X(II,K) - X(IM,K)
            X(II,K) = X(II,K) + X(IM,K)
            X(IM,K) = CT * CX
130   CONTINUE
      L = 2 * L
      I1 = I2
      IF (L .LE. M2)  GOTO 120
C
      DO 150 I = 1, M
        II = IP(I1,I)
        IF (II .GT. I)  THEN
          DO 140 K = 1, N
            CT = X(I,K)
            X(I,K) = X(II,K)
            X(II,K) = CT
140       CONTINUE
        ENDIF
150   CONTINUE
C
      RETURN
      END
C
      SUBROUTINE CFFT2D2 (IS, M, M1, N, X, W, IP)
C
C     PERFORMS COMPLEX RADIX 2 FFTS ON THE SECOND DIMENSION OF THE 2-D ARRAY X
C     D H BAILEY  11/15/84
C
      COMPLEX X(M1,N), W(N), CT, CX
      INTEGER IP(2,N)
      DATA PI/3.14159265358979 3/
C
C     IF  IS = 0  THEN INITIALIZE ONLY
C
      N2 = N / 2
      IF (IS .EQ. 0)  THEN
        DO 100 I = 1, N2
          T = 2. * PI * (I-1) / N
          W(I) = CMPLX (COS (T), SIN (T))
100     CONTINUE
        RETURN
      ENDIF
C
C     PERFORM FORWARD OR BACKWARD FFTS ACCORDING TO  IS = 1 OR -1
C
      DO 110 I = 1, N
        IP(1,I) = I
110   CONTINUE
      L = 1
      I1 = 1
C
120   I2 = 3 - I1
      DO 130 J = L, N2, L
        CX = W(J-L+1)
        IF (IS .LT. 0)  CX = CONJG (CX)
        DO 130 I = J-L+1, J
          II = IP(I1,I)
          IP(I2,I+J-L) = II
          IM = IP(I1,I+N2)
          IP(I2,I+J) = IM
          DO 130 K = 1, M
            CT = X(K,II) - X(K,IM)
            X(K,II) = X(K,II) + X(K,IM)
            X(K,IM) = CT * CX
130   CONTINUE
      L = 2 * L
      I1 = I2
      IF (L .LE. N2)  GOTO 120
C
      DO 150 I = 1, N
        II = IP(I1,I)
        IF (II .GT. I)  THEN
          DO 140 K = 1, M
            CT = X(K,I)
            X(K,I) = X(K,II)
            X(K,II) = CT
```

```
140       CONTINUE
          ENDIF
150   CONTINUE
C
      RETURN
      END
C
      SUBROUTINE CHOTST (ER, FP, TM)
C
C     CHOLSKY TEST PROGRAM
C
      PARAMETER (IDA=250, NMAT=250, M=4, N=40, NRHS=3, F7=78125.,
     $ T30=1073741824.)
      COMMON /ARRAYS/ A(0:IDA, -M:0, 0:N), B(0:NRHS, 0:NMAT, 0:N),
     $    AX(0:IDA, -M:0, 0:N), BX(0:NRHS, 0:NMAT, 0:N)
      DATA  IT/200/, ANS/5177.885317745621
C
C     INITIALIZE
C
      LA = (IDA+1) * (M+1) * (N+1)
      LB = (NRHS+1) * (NMAT+1) * (N+1)
      T = F7 / T30
      DO 100 K = 0, N
        DO 100 J = -M, 0
          DO 100 I = 0, IDA
            T = MOD (F7 * T, 1.)
            AX(I,J,K) = T
100     CONTINUE
      DO 110 K = 0, N
        DO 110 J = 0, NMAT
          DO 110 I = 0, NRHS
            T = MOD (F7 * T, 1.)
            BX(I,J,K) = T
110   CONTINUE
      TM = CPTIME ()
C
C     BEGIN TIMING TEST
C
      DO 120 J = 1, IT
        CALL COPY (LA, AX, A)
        CALL COPY (LB, BX, B)
        CALL CHOLSKY (IDA, NMAT, M, N, A, NRHS, IDA, B)
120   CONTINUE
C
      TM = CPTIME ()
      ER = ABS ((B(1,19,19) - ANS) / ANS)
      FP = IT * (NMAT + 1) * 4483.
C
      RETURN
      END
C
      SUBROUTINE CHOLSKY (IDA, NMAT, M, N, A, NRHS, IDB, B)
C
C     CHOLESKY DECOMPOSITION/SUBSTITUTION SUBROUTINE.
C
C     11/28/84  D H BAILEY  MODIFIED FOR NAS KERNEL TEST
C
      REAL A(0:IDA, -M:0, 0:N), B(0:NRHS, 0:IDB, 0:N), EPSS(0:256)
      DATA EPS/1E-13/
C
C     CHOLESKY DECOMPOSITION
C
      DO 1 J = 0, N
        I0 = MAX ( -M, -J )
C
C     OFF DIAGONAL ELEMENTS
C
        DO 2 I = I0, -1
          DO 3 JJ = I0 - I, -1
            DO 3 L = 0, NMAT
3             A(L,I,J) = A(L,I,J) - A(L,JJ,I+J) * A(L,I+JJ,J)
          DO 2 L = 0, NMAT
2           A(L,I,J) = A(L,I,J) * A(L,0,I+J)
C
C     STORE INVERSE OF DIAGONAL ELEMENTS
C
        DO 4 L = 0, NMAT
4         EPSS(L) = EPS * A(L,0,J)
        DO 5 JJ = I0, -1
          DO 5 L = 0, NMAT
5           A(L,0,J) = A(L,0,J) - A(L,JJ,J) ** 2
        DO 1 L = 0, NMAT
1         A(L,0,J) = 1. / SQRT ( ABS (EPSS(L) + A(L,0,J)) )
C
C     SOLUTION
C
      DO 6 I = 0, NRHS
        DO 7 K = 0, N
          DO 8 L = 0, NMAT
8           B(I,L,K) = B(I,L,K) * A(L,0,K)
          DO 7 JJ = 1, MIN (M, N-K)
            DO 7 L = 0, NMAT
7             B(I,L,K+JJ) = B(I,L,K+JJ) -  A(L,-JJ,K+JJ) * B(I,L,K)
C
        DO 6 K = N, 0, -1
          DO 9 L = 0, NMAT
9           B(I,L,K) = B(I,L,K) * A(L,0,K)
          DO 6 JJ = 1, MIN (M, K)
            DO 6 L = 0, NMAT
6             B(I,L,K-JJ) = B(I,L,K-JJ) - A(L,-JJ,K) * B(I,L,K)
C
      RETURN
      END
C
      SUBROUTINE BTRTST (ER, FP, TM)
C
C     BTRIX TEST PROGRAM
C
      PARAMETER (JD=30, KD=30, LD=30, MD=30, F7=78125., T30=1073741824.)
      COMMON /ARRAYS/ S(JD,KD,LD,5), A(5,5,MD,MD), B(5,5,MD,MD),
     $     C(5,5,MD,MD), SX(JD,KD,LD,5), BX(5,5,MD,MD)
      DATA JS/2/, JE/29/, LS/2/, LE/29/, IT/20/, ANS/-0.286282658663962/
C
C     INITIALIZATION
C
      NB = 25 * MD * MD
      NS = JD * KD * LD * 5
      T = F7 / T30
      DO 100 L = 1, MD
        DO 100 K = 1, MD
          DO 100 J = 1, 5
            DO 100 I = 1, 5
              T = MOD (F7 * T, 1.)
              A(I,J,K,L) = T
              T = MOD (F7 * T, 1.)
              BX(I,J,K,L) = T
              T = MOD (F7 * T, 1.)
              C(I,J,K,L) = T
100   CONTINUE
      DO 110 L = 1, 5
        DO 110 K = 1, LD
          DO 110 J = 1, KD
            DO 110 I = 1, JD
              T = MOD (F7 * T, 1.)
              SX(I,J,K,L) = T
110   CONTINUE
      TM = CPTIME ()
C
C     TIMING TEST
C
      DO 120 II = 1, IT
        CALL COPY (NS, SX, S)
        DO 120 K = 1, KD
          CALL COPY (NB, BX, B)
          CALL BTRIX (JS, JE, LS, LE, K)
120   CONTINUE
```

```
C
      TM = CPTIME ()
      ER = ABS ((S(19,19,19,1) - ANS) / ANS)
      FP = IT * MD * (LE - 1) * 1916S.
C
      RETURN
      END
C
      SUBROUTINE BTRIX (JS, JE, LS, LE, K)
C
C     VECTORIZED BLOCK TRI-DIAGONAL SOLVER IN THE J DIRECTION
C         FOR K = CONSTANT PLANES
C
C   11/15/84  D H BAILEY  MODIFIED FOR NAS KERNEL TEST
C
      PARAMETER (JD=30, KD=30, LD=30, MD=30)
      COMMON /ARRAYS/ S(JD,KD,LD,5), A(5,5,MD,MD), B(5,5,MD,MD),
     $     C(5,5,MD,MD)
C
      DIMENSION    U12(MD), U13(MD), U14(MD), U15(MD), U23(MD),
     $             U24(MD), U25(MD), U34(MD), U35(MD), U45(MD)
C
      REAL         L11(MD), L21(MD), L31(MD), L41(MD), L51(MD),
     $             L22(MD), L32(MD), L42(MD), L52(MD), L33(MD),
     $             L43(MD), L53(MD), L44(MD), L54(MD), L55(MD)
C
C     PART 1.  FORWARD BLOCK SWEEP
C
C
      DO 100 J   = JS,JE
C
C**********   STEP 1.  CONSTRUCT L(I) IN B    **************************
C
      IF(J.EQ.JS) GO TO 4
      DO 3 M = 1,5
        DO 3 N = 1,5
          DO 3 L = LS,LE
            B(M,N,J,L) = B(M,N,J,L) - A(M,1,J,L)*B(1,N,J-1,L)
     $           - A(M,2,J,L)*B(2,N,J-1,L) - A(M,3,J,L)*B(3,N,J-1,L)
     $           - A(M,4,J,L)*B(4,N,J-1,L) - A(M,5,J,L)*B(5,N,J-1,L)
    3 CONTINUE
C
    4 CONTINUE
C
C**********   STEP 2.  COMPUTE L INVERSE    **************************
C
C
C       A.  DECOMPOSE L(I) INTO L AND U
C
      DO 20 L = LS,LE
        L11(L)  = 1. / B(1,1,J,L)
        U12(L)  = B(1,2,J,L)*L11(L)
        U13(L)  = B(1,3,J,L)*L11(L)
        U14(L)  = B(1,4,J,L)*L11(L)
        U15(L)  = B(1,5,J,L)*L11(L)
        L21(L)  = B(2,1,J,L)
        L22(L)  = 1. / (B(2,2,J,L) - L21(L)*U12(L))
        U23(L)  = (B(2,3,J,L) - L21(L)*U13(L)) * L22(L)
        U24(L)  = (B(2,4,J,L) - L21(L)*U14(L)) * L22(L)
        U25(L)  = (B(2,5,J,L) - L21(L)*U15(L)) * L22(L)
        L31(L)  = B(3,1,J,L)
        L32(L)  = B(3,2,J,L) - L31(L)*U12(L)
        L33(L)  = 1. / (B(3,3,J,L) - L31(L)*U13(L) - L32(L)*U23(L))
        U34(L)  = (B(3,4,J,L) - L31(L)*U14(L) - L32(L)*U24(L)) * L33(L)
        U35(L)  = (B(3,5,J,L) - L31(L)*U15(L) - L32(L)*U25(L)) * L33(L)
   20 CONTINUE
C
      DO 25 L = LS,LE
        L41(L)  = B(4,1,J,L)
        L42(L)  = B(4,2,J,L) - L41(L)*U12(L)
        L43(L)  = B(4,3,J,L) - L41(L)*U13(L) - L42(L)*U23(L)
        L44(L)  = 1. / (B(4,4,J,L) - L41(L)*U14(L) - L42(L)*U24(L)
     $      - L43(L)*U34(L))
        U45(L)  = (B(4,5,J,L) - L41(L)*U15(L) - L42(L)*U25(L)
     $      - L43(L)*U35(L)) * L44(L)
        L51(L)  = B(5,1,J,L)
        L52(L)  = B(5,2,J,L) - L51(L)*U12(L)
        L53(L)  = B(5,3,J,L) - L51(L)*U13(L) - L52(L)*U23(L)
        L54(L)  = B(5,4,J,L) - L51(L)*U14(L) - L52(L)*U24(L)
     $      - L53(L)*U34(L)
        L55(L)  = 1. / (B(5,5,J,L) - L51(L)*U15(L) - L52(L)*U25(L)
     $      - L53(L)*U35(L) - L54(L)*U45(L))
   25 CONTINUE
C
C**********   STEP 3.  SOLVE FOR INTERMEDIATE VECTOR    **************************
C
C       A.  CONSTRUCT RHS
C
      IF(J.EQ.JS) GO TO 34
      DO 33 M = 1,5
        DO 33 L = LS,LE
          S(J,K,L,M) = S(J,K,L,M) - A(M,1,J,L)*S(J-1,K,L,1)
     $         - A(M,2,J,L)*S(J-1,K,L,2) - A(M,3,J,L)*S(J-1,K,L,3)
     $         - A(M,4,J,L)*S(J-1,K,L,4) - A(M,5,J,L)*S(J-1,K,L,5)
   33 CONTINUE
C
C       B. INTERMEDIATE VECTOR
C
   34 CONTINUE
C
C       FWD SUBSTITUTION
C
      DO 35 L = LS,LE
        D1 = S(J,K,L,1)*L11(L)
        D2 = (S(J,K,L,2) - L21(L)*D1) * L22(L)
        D3 = (S(J,K,L,3) - L31(L)*D1 - L32(L)*D2) * L33(L)
        D4 = (S(J,K,L,4) - L41(L)*D1 - L42(L)*D2 - L43(L)*D3) * L44(L)
        D5 = (S(J,K,L,5) - L51(L)*D1 - L52(L)*D2 - L53(L)*D3
     $       - L54(L)*D4) * L55(L)
C
C       BWD SUBSTITUTION
C
        S(J,K,L,5)  = D5
        S(J,K,L,4)  = D4 - U45(L)*D5
        S(J,K,L,3)  = D3 - U34(L)*S(J,K,L,4) - U35(L)*D5
        S(J,K,L,2)  = D2 - U23(L)*S(J,K,L,3) - U24(L)*S(J,K,L,4)
     $       - U25(L)*D5
        S(J,K,L,1)  = D1 - U12(L)*S(J,K,L,2) - U13(L)*S(J,K,L,3)
     $       - U14(L)*S(J,K,L,4) - U15(L)*D5
   35 CONTINUE
C
C**********   STEP 4.  CONSTRUCT U(I) = L(I)**(-1)*C(I+1)    **********
C**********            BY COLUMNS AND STORE IN B             **********
C
      IF(J.EQ.JE) GO TO 100
      DO 40 N = 1,5
        DO 40 L = LS,LE
C
C       FWD SUBSTITUTION
C
          C1 = C(1,N,J,L)*L11(L)
          C2 = (C(2,N,J,L) - L21(L)*C1) * L22(L)
          C3 = (C(3,N,J,L) - L31(L)*C1 - L32(L)*C2) * L33(L)
          C4 = (C(4,N,J,L) - L41(L)*C1 - L42(L)*C2 - L43(L)*C3)
     $         * L44(L)
          C5 = (C(5,N,J,L) - L51(L)*C1 - L52(L)*C2 - L53(L)*C3
     $         - L54(L)*C4) * L55(L)
C
C       BWD SUBSTITUTION
C
          B(5,N,J,L)  = C5
          B(4,N,J,L)  = C4 - U45(L)*C5
          B(3,N,J,L)  = C3 - U34(L)*B(4,N,J,L) - U35(L)*C5
          B(2,N,J,L)  = C2 - U23(L)*B(3,N,J,L) - U24(L)*B(4,N,J,L)
     $         - U25(L)*C5
          B(1,N,J,L)  = C1 - U12(L)*B(2,N,J,L) - U13(L)*B(3,N,J,L)
```

```
     $        - U14(L)*B(4,N,J,L) - U15(L)*C5
   40 CONTINUE
C
C
  100 CONTINUE
C
C     PART 2.  BACKWARD BLOCK SWEEP
C
      JEM1 = JE - 1
C
      DO 200 J = JEM1,JS,-1
        DO 200 M = 1,5
          DO 200 L = LS,LE
            S(J,K,L,M) = S(J,K,L,M) - B(M,1,J,L)*S(J+1,K,L,1)
     $           - B(M,2,J,L)*S(J+1,K,L,2) - B(M,3,J,L)*S(J+1,K,L,3)
     $           - B(M,4,J,L)*S(J+1,K,L,4) - B(M,5,J,L)*S(J+1,K,L,5)
  200 CONTINUE
C
      RETURN
      END
C
      SUBROUTINE GMTTST (ER, FP, TM)
C
      PARAMETER (NW=100, NB=5, F7=78125., T30=1073741824.)
      COMPLEX WALL, ZCR, PROJ, ZI, Z1, ZZ
      COMMON /ARRAYS/ NWALL(NB), WALL(NW,NB), RMATRX(NW*NB,NW*NB),
     $ ZCR(NW,NB), PROJ(NW,NB), XMAX(NB)
      DATA IT/2/, ANS/-2.57754233214174/
C
C   INITIALIZATION
C
      LW = 2 * NW * NB
      T2 = F7 / T30
      DO 100 J = 1, NB
        NWALL(J) = NW
  100 CONTINUE
      DO 110 J = 1, NB
        DO 110 I = 1, NW
          T1 = MOD (F7 * T2, 1.)
          T2 = MOD (F7 * T1, 1.)
          WALL(I,J) = CMPLX (T1, T2)
  110 CONTINUE
      TM = CPTIME ()
C
C   TIMING TEST
C
      DO 120 I = 1, IT
        CALL GMTRY
  120 CONTINUE
C
      TM = CPTIME () - TM
      ER = ABS ((RMATRX(19,19) - ANS) / ANS)
      FP = IT * (120. * (NB*NW) ** 2  +  0.666 * (NB*NW) ** 3)
C
      RETURN
      END
C
      SUBROUTINE GMTRY
C
C  COMPUTE SOLID-RELATED ARRAYS.  GAUSS ELIMINATE THE MATRIX OF WALL
C  INFLUENCE COEFFICIENTS.
C
C    11/30/84  D H BAILEY    REVISED CODE FOR NAS KERNEL TEST
C
      PARAMETER (NW=100, NB=5)
      COMPLEX WALL, ZCR, PROJ, ZI, Z1, ZZ
      COMMON /ARRAYS/ NWALL(NB), WALL(NW,NB), RMATRX(NW*NB,NW*NB),
     $ ZCR(NW,NB), PROJ(NW,NB), XMAX(NB)
C
      DATA  ARCL /5./, PI /3.14159265358979323/, PERIOD/3./
C
C  COMPUTE ARCLENGTH.
C
      MATDIM = 0
      ARCL = 0.
      YMIN = 1.E+50
      YMAX = -1.E+50
      PIDP = PI / PERIOD
C
      DO 9 L = 1, NB
        MATDIM = MATDIM + NWALL(L)
        DO 9 K = 1,NWALL(L)
          ARCL = ARCL + ABS(WALL(K,L) - WALL(1+MOD(K,NWALL(L)), L))
    9 CONTINUE
C
C  COMPUTE CORE RADIUS.
C
      R0 = ARCL / (MATDIM*2.)
      SIGMA = R0 / 2.
C
C  DEFINE CREATION POINTS.
C
      DO 6 L = 1,NB
        DO 5 K = 1,NWALL(L)
          ZZ = WALL(1+MOD(K+NWALL(L)-2,NWALL(L)), L)
     &         - WALL(1+MOD(K,NWALL(L)), L)
          ZCR(K,L) = WALL(K,L) + CMPLX(0., R0/ABS(ZZ)) * ZZ
    5   CONTINUE
C
C  CHECK THAT WALL AND CREATION POINTS ARE NOT CROSSED DUE TO
C  TOO SHARP A CONCAVE KINK OR AN ERROR IN DEFINING THE BODY.
C  ALSO FIND HIGHEST,  LOWEST AND RIGHT-MOST POINT.
C
        XMAX(L) = REAL(ZCR(1,L))
        LS = 0
        DO 6 K = 1,NWALL(L)
          YMIN = MIN (YMIN, AIMAG(ZCR(K,L)))
          YMAX = MAX (YMAX, AIMAG(ZCR(K,L)))
          XMAX(L) = MAX (XMAX(L), REAL(ZCR(K,L)))
          KP = 1 + MOD(K, NWALL(L))
          IF (REAL((ZCR(KP,L) - ZCR(K,L)) *
     &        CONJG(WALL(KP,L) - WALL(K,L))).GT.0.) THEN
            LS = L
            KS = K
          ENDIF
    6   CONTINUE
C
C       IF (LS .NE. 0) THEN
C         WRITE (6, 102) LS, KS
C102      FORMAT(" ON BODY NUMBER ", I3, " YOU HAVE TOO SHARP A",
C     &       " KINK NEAR POINT ", I4)
C         STOP
C       ENDIF
C
C  THE "MAIN PERIOD" WILL BE BETWEEN YLIMIT AND YLIMIT + PERIOD.
C
      YLIMIT = (YMIN - PERIOD + YMAX)/2
C
C  PROJECT CREATION POINTS INTO MAIN PERIOD. THIS IS TECHNICAL.
C
      DO 1 L = 1,NB
        DO 1 K = 1,NWALL(L)
          PROJ(K,L) = ZCR(K,L) - CMPLX(0., PERIOD*
     &          (INT(5. + (AIMAG(ZCR(K,L)) - YLIMIT) / PERIOD) - 5.))
    1   CONTINUE
C
C   COMPUTE MATRIX.
C
      SIG2 = (2. * PIDP * SIGMA) ** 2
      I0 = 0
      DO 2 L1 = 1,NB
        J0 = 0
        DO 4 L2 = 1,NB
          KRON = 0
          IF (L1 .EQ. L2) KRON = 1
          DO 3 J = 1,NWALL(L2)
```

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-86711		

4. Title and Subtitle	5. Report Date
THE NAS KERNEL BENCHMARK PROGRAM	July 1985
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
David H. Bailey and John T. Barton	85195
	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
Ames Research Center Moffett Field, CA 94035	
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Memorandum
National Aeronautics and Space Administration Washington, DC 20546	14. Sponsoring Agency Code 536-01-11

15. Supplementary Notes

Point of contact: John T. Barton, Ames Research Center, MS 233-1, Moffett Field, CA 94035, (415)694-6837 or FTS 464-6837

16. Abstract

A collection of benchmark test kernels that measure supercomputer performance has been developed for the use of the NAS (Numerical Aerodynamic Simulation) program at the NASA Ames Research Center. This article describes this benchmark program in detail and gives the specific ground rules for running the program as a performance test.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Supercomputers Benchmark	Unlimited Subject Category - 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	20	

*For sale by the National Technical Information Service, Springfield, Virginia 22161

End of Document